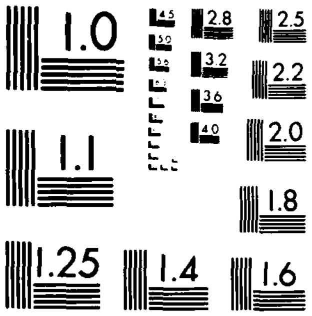
AD-A189 549 ADA (TRADE NAME) FROM A MANAGEMENT PERSPECTIVE FOR HIGH-LEVEL SECRETARIAT AND STAFF(U) ADA JOINT PROGRAM OFFICE ARLINGTON VA C ENGLE ET AL. 03 DEC 86 1/1

UNCLASSIFIED F/G 12/5 NL

END
DATE
FILMED
3 88


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

DTIC FILE COPY

REPORT DOCUMENTATION PAGE

AD-A189 549

	READ INSTRUCTIONS BEFORE COMPLETING FORM
2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Ada from a Management Perspective for High-Level Secretariat and Staff	5. TYPE OF REPORT & PERIOD COVERED Tutorial, 3 Dec., 1986
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) MAJ Charles Engle, and LT Anthony Dominice	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION AND ADDRESS Ada Software Education and Training Team Ada Joint Program Office, 3E114, The Pentagon, Washington, D.C.20301-3081	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Ada Joint Program Office 3E 114, The Pentagon Washington, DC 20301-3081	12. REPORT DATE 3 December, 1986
	13. NUMBER OF PAGES 89
14. MONITORING AGENCY NAME & ADDRESS(*If different from Controlling Office*) Ada Joint Program Office	15. SECURITY CLASS (*of this report*) UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (*of this Report*)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (*of the abstract entered in Block 20. If different from Report*)

UNCLASSIFIED

DTIC ELECTE JAN 0 6 1988 S D

D

18. SUPPLEMENTARY NOTES

19. KEYWORDS (*Continue on reverse side if necessary and identify by block number*)

Ada Programming language, Ada Training, Education, Training, Computer Programs, Ada Joint Program Office, AJPO

20. ABSTRACT (*Continue on reverse side if necessary and identify by block number*)

This document contains prints of viewgraphs presented as an introductory tutorial on Ada on 3 December, 1986.

DD FORM 1473 EDITION OF 1 NOV 65 IS OBSOLETE
1 JAN 73 S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

Ada®
FROM A MANAGEMENT PERSPECTIVE FOR HIGH–LEVEL SECRETARIAT AND STAFF

MAJOR CHARLES ENGLE: UNITED STATES MILTARY ACADEMY
WEST POINT, N.Y.

1LT ANTHONY DOMINICE: KEESLER TECH TRAINING CENTER
KEESLER AFB, MS.

3 DECEMBER 1986

SPONSORED BY:

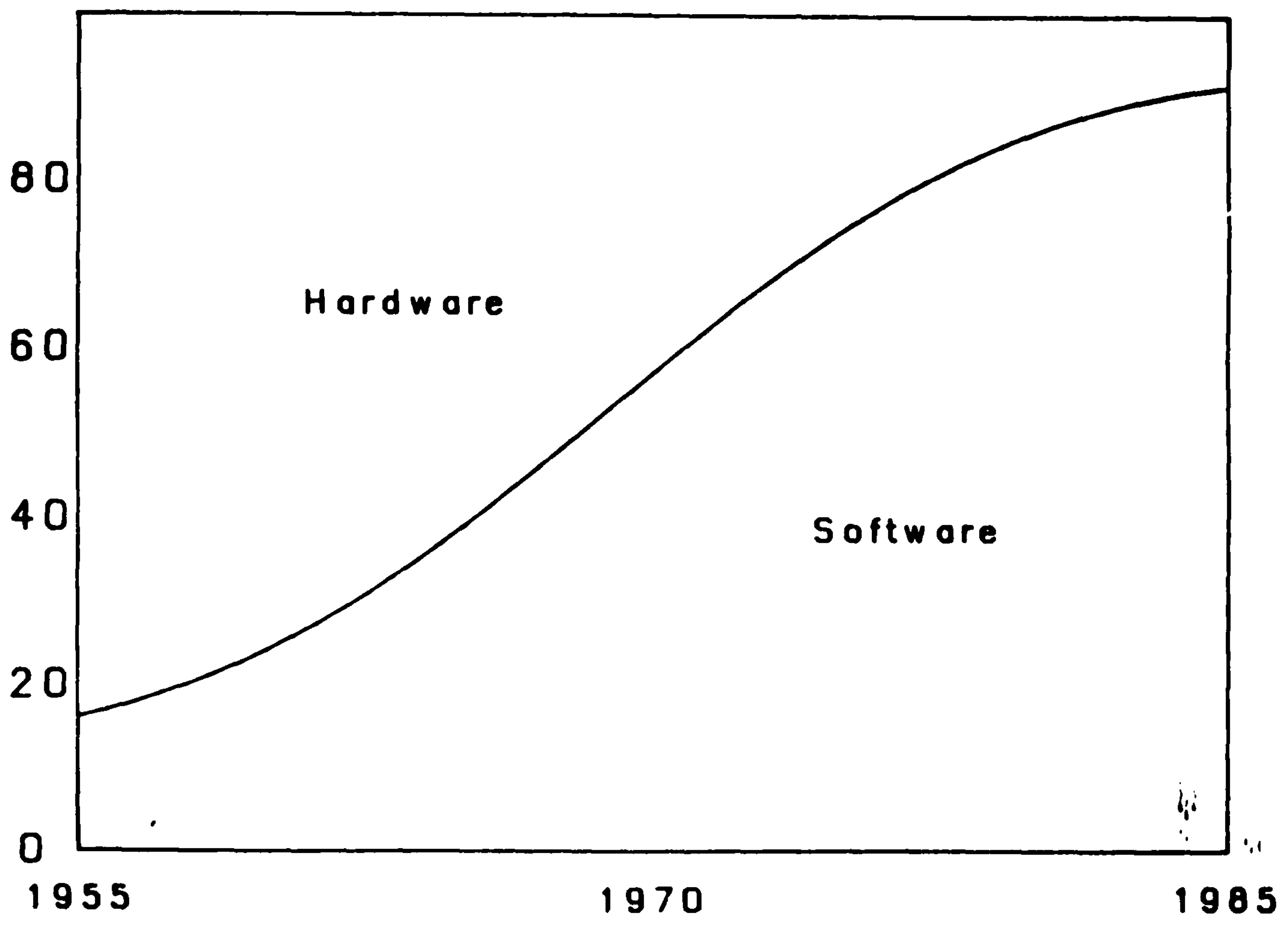
Ada JOINT PROGRAM OFFICE (AJPO)

Ada SOFTWARE ENGINEERING EDUCATION AND TRAINING (ASEET) TEAM

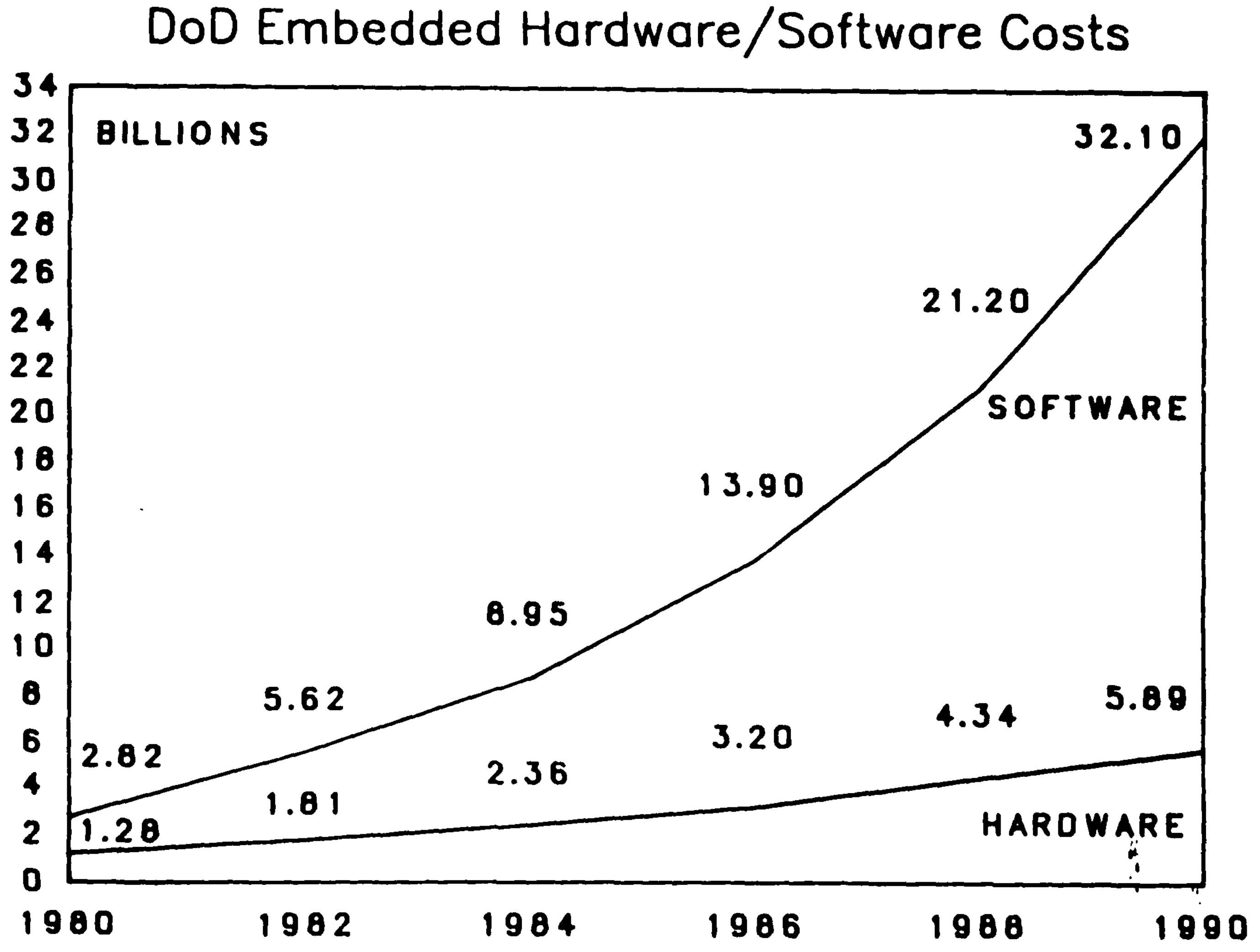
WHAT YOU MAY HAVE HEARD ABOUT Ada

* It's a cure—all for DoD computing
* It's just another D—— acronym
* It's a programming language
* It's "just another programming language"
* Life cycle costs, support environments, STARS, Methodologies, SEI ?? !! It's everything

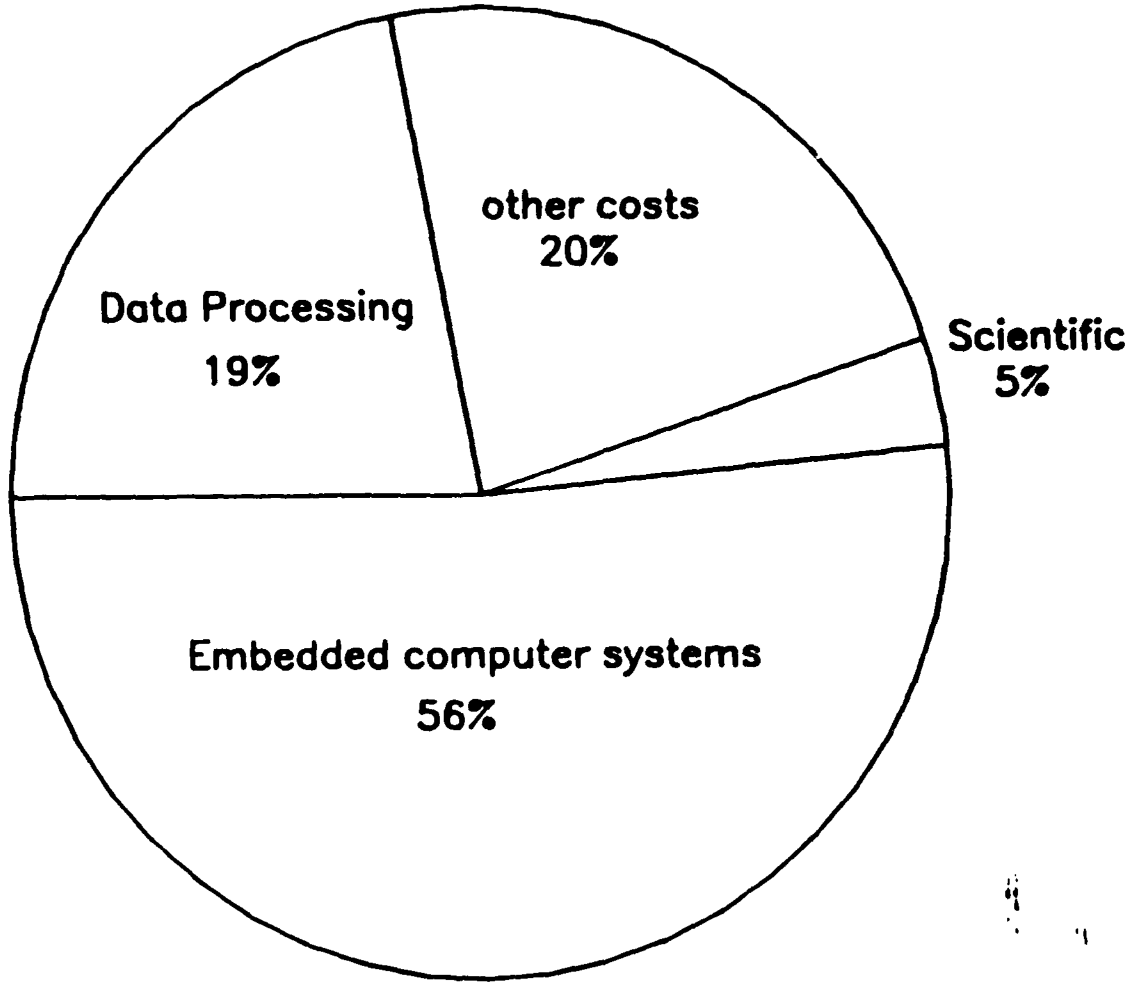
Software Crisis



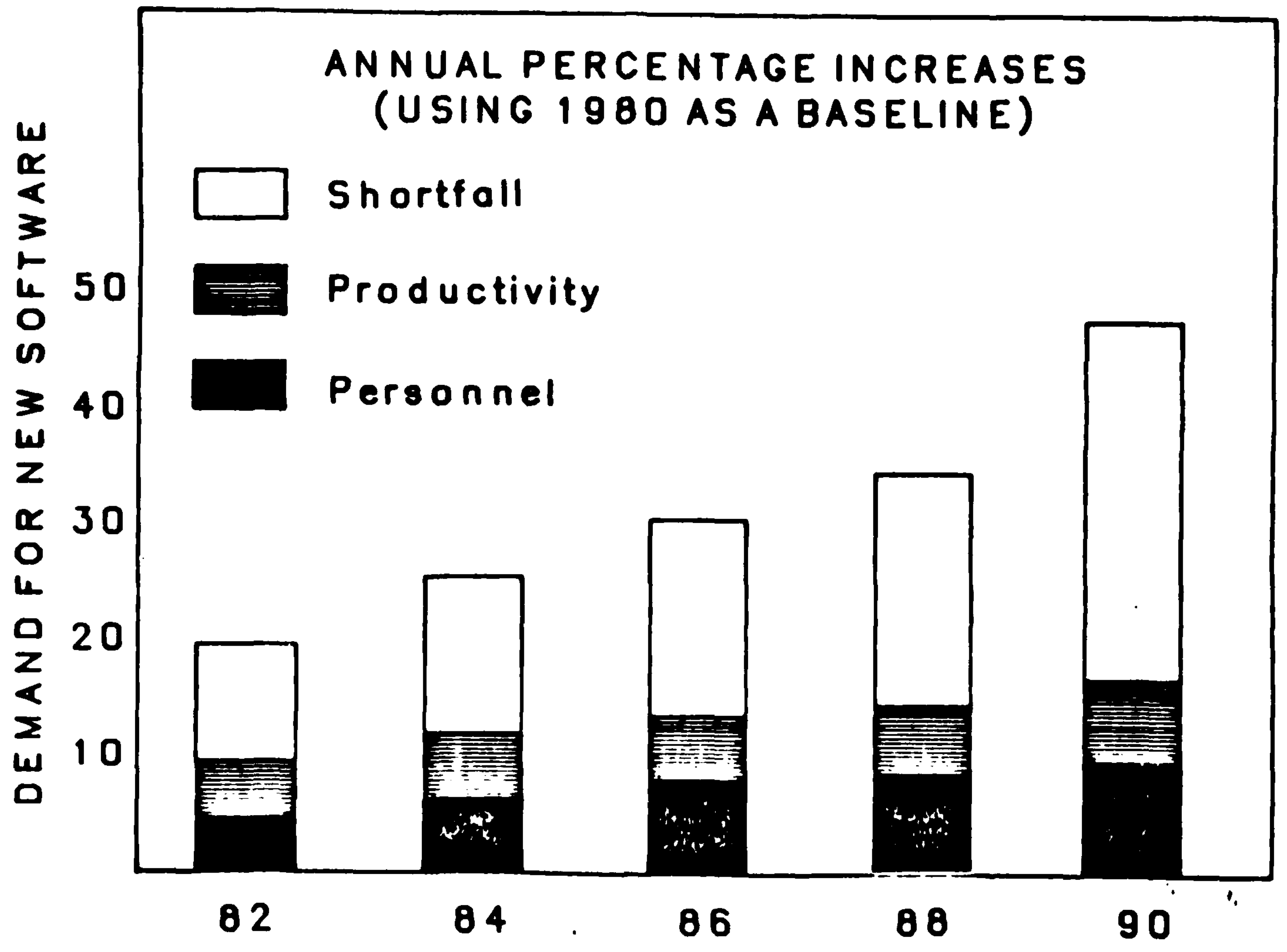
Software Crisis



Software Crisis



Software Crisis



WHAT YOU NEED TO HEAR ABOUT Ada

Plain and simple ...

* Ada is a standardized computer programming language developed by the DoD for use in embedded computer systems

* Ada is the BEST tool available for meeting the software engineering requirements of the DoD

OVERVIEW

* Rationale for development
* Capabilities and advantages
* Life Cycle application

OVERVIEW

* Rationale for development
* Capabilities and advantages
* Life Cycle application

THE CRITICALITY OF SOFTWARE

* Hardware is no longer the dominant factor in the hardware/software relationship
    - Cost
    - Technology
* The demand for software is rising exponentially
* The cost of software is rising exponentially
* Software maintenance is the dominant software activity
* Systems are getting more complex
* Life and property are dependent on software

CHARACTERISTICS OF DoD SOFTWARE

* Expensive
* Incorrect
* Unreliable
* Difficult to predict
* Unmaintainable
* Not reusable

FACTORS AFFECTING DoD SOFTWARE

* Ignorance of life cycle implications
* Lack of standards
* Lack of methodologies
* Inadequate support tools
* Management
* Software professionals

CHARACTERISTICS OF DoD SOFTWARE REQUIREMENTS

* Large
* Complex
* Long lived
* High reliability
* Time constraints
* Size constraints

TRADITIONAL APPROACH TO SOFTWARE

* A necessary evil

* A black art

* Guru's and magicians in a dark
  room

(with due respect to sofware professionals)

THE FUNDAMENTAL PROBLEM

* Our inability to manage the COMPLEXITY of our software systems

* Lack of a disciplined, engineering approach

SOFTWARE ENGINEERING

THE ESTABLISHMENT AND APPLICATION OF SOUND ENGINEERING ==>

* Environments
* Tools
* Methodologies
* Models
* Principles
* Concepts

SOFTWARE ENGINEERING

COMBINED WITH =>

* Standards
* Guidelines
* Practices

SOFTWARE ENGINEERING

TO SUPPORT COMPUTING WHICH IS =>

* Understandable
* Efficient
* Reliable and safe
* Modifiable
* Correct

THROUGHOUT THE LIFE CYCLE OF A SYSTEM

(C. McKAY, 1985)

PROGRAMMING LANGUAGES AND SOFTWARE ENGINEERING

* A programming language is a software engineering tool

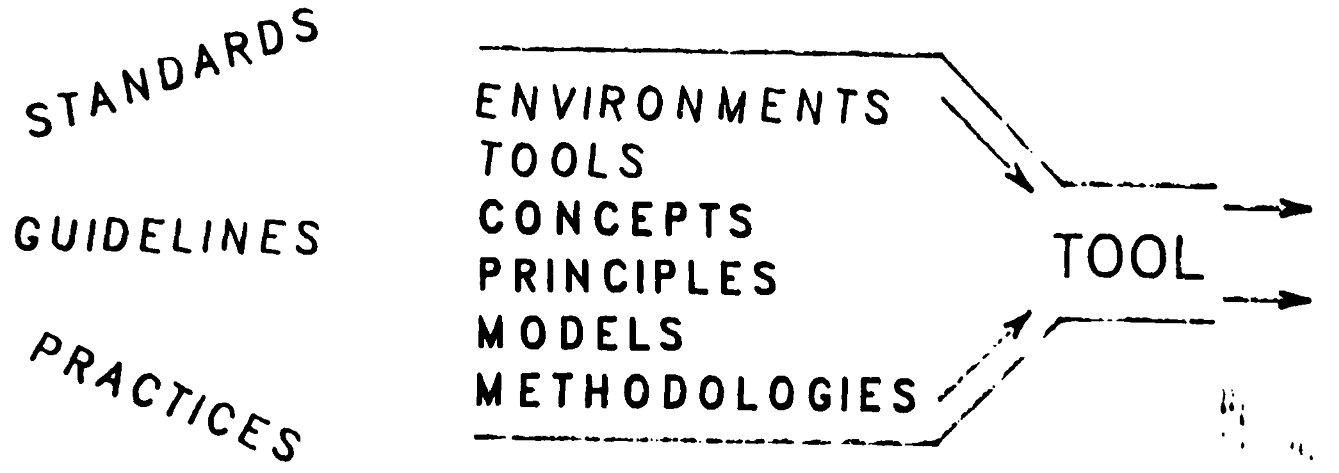
* A programming language EXPRESSES and EXECUTES design methodologies

* The quality of a programming language for software engineering is determined by how well it supports a design methodology and its underlying models, principles, and concepts.

TRADITIONAL PROGRAMMING LANGUAGES AND SOFTWARE ENGINEERING

Programming Languages

* Were not engineered
* Have lacked the ability to express good software engineering
* Have acted to constrain software engineering



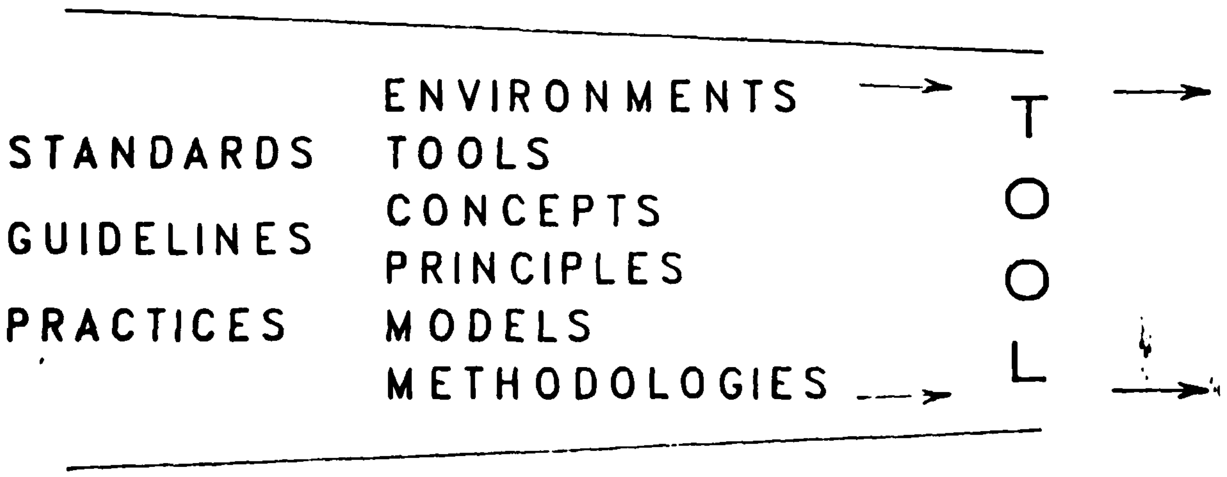
Ada
AND
SOFTWARE ENGINEERING

Ada

* Was itself "engineered" to support software engineering

* Embodies the same concepts, principles, and models to support methodologies

* Is the best tool (programming language) for software engineering currently available



PRINCIPLES OF SOFTWARE ENGINEERING

* Abstraction
* Modularity
* Localization
* Information hiding
* Completeness
* Confirmability
* Uniformity

ABSTRACTION

* The process of separating out the important parts of something while ignoring the inessential details

* Separates the "what" from the "how"

* Reduces the level of complexity

* There are levels of abstraction within a system

MODULARITY

* Purposeful structuring of a system into parts which work together

* Each part performs some smaller task of the overall system

* Can concentrate and develop parts independently as long as interfaces are defined and shared

* Can develop hierarchies of management and implementation

LOCALIZATION

* Putting things that logically belong together in the same physical place

INFORMATION HIDING

* Puts a wall around localized details

* Prevents reliance upon details and causes focus of attention to interfaces and logical properties

COMPLETENESS

* Ensuring all important parts are present
* Nothing left out

CONFIRMABILITY

* Developing parts that can be effectively tested

UNIFORMITY

* No unnecessary differences across a system

OVERVIEW

* Rationale for development
* Capabilities and advantages
* Life Cycle application

MAJOR FEATURES OF Ada

* Standardization
* Readability
* Program Units
* Separate Compilation
* Subprograms
* Packages
* Strong Typing
* Typing Structures
* Data Abstraction
* Tasks
* Exceptions
* Generics

MAJOR FEATURES OF Ada

* Standardization
* Readability
* Program Units
* Separate Compilation
* Subprograms
* Packages

* Strong Typing
* Typing Structures
* Data Abstraction
* Tasks
* Exceptions
* Generics

STANDARDIZATION

* Ada is an exact standard
  - ANSI/MIL–STD–1815A
  - No subsets, no supersets
* Conformance to the standard is required
  - Trademark control
  - Ada Compiler Validation Capability (ACVC)
* andardization allows for portability
* Standardization promotes reusability
* Standardization shifts focus from the mundane to the important

MAJOR FEATURES OF Ada

* Standardization
* $\boxed{\text{Readability}}$
* Program Units
* Separate Compilation
* Subprograms
* Packages

* Strong Typing
* Typing Structures
* Data Abstraction
* Tasks
* Exceptions
* Generics

READABILITY

* Ada was engineered with the understanding that programming is a human activity

* Features are provided that allow a maintenance person to quickly grasp the meaning of a particular program and to understand its structure

* Readability is more than just a language issue

MAJOR FEATURES OF Ada

* Standardization
* Readability
* $\boxed{\text{* Program Units}}$
* Separate Compilation
* Subprograms
* Packages
* Strong Typing
* Typing Structures
* Data Abstraction
* Tasks
* Exceptions
* Generics

SYSTEMS ENGINEERING

* Analyze problem
* Break into solvable parts
* Implement parts
* Test parts
* Integrate parts to form total system
* Test total system

REQUIREMENTS FOR EFFECTIVE SYSTEMS ENGINEERING

* Ability to express architecture
* Ability to define and enforce interfaces
* Ability to create independent components
* Ability to separate architecture issues from implementation issues

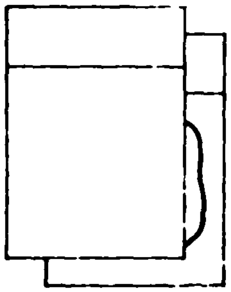
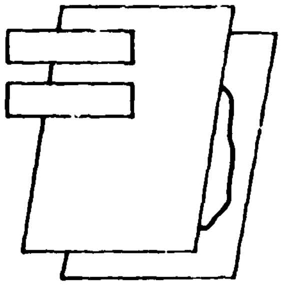
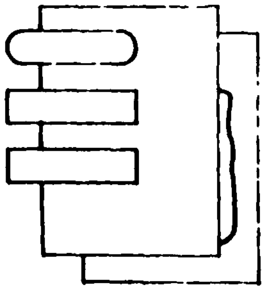
PROGRAM UNITS

* Components of Ada which together form a working Ada software system

* Express the architecture of a system

* Define and enforce interfaces

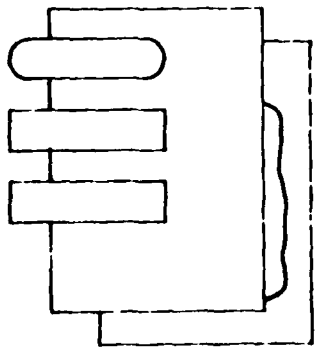
PROGRAM UNITS

	SUBPROGRAMS	Working components that perform some action
	TASKS	Performs actions in parallel with other program units
	PACKAGES	A mechanism for collecting entities together into logical units

PROGRAM UNITS

* Consist of two parts: specification and body

SPECIFICATION: Defines the interface between the program unit and other program units (the **WHAT**)



BODY: Defines the implementation of the program unit (the **HOW**)

PROGRAM UNITS

* The specification of the program unit is the only means of connecting program units

* The interface is enforced

* The body of a program unit is not accessible to other program units

* There is a clear distinction between architecture and implementation

MAJOR FEATURES OF Ada

* Standardization
* Readability
* Program Units
* Separate Compilation
* Subprograms
* Packages

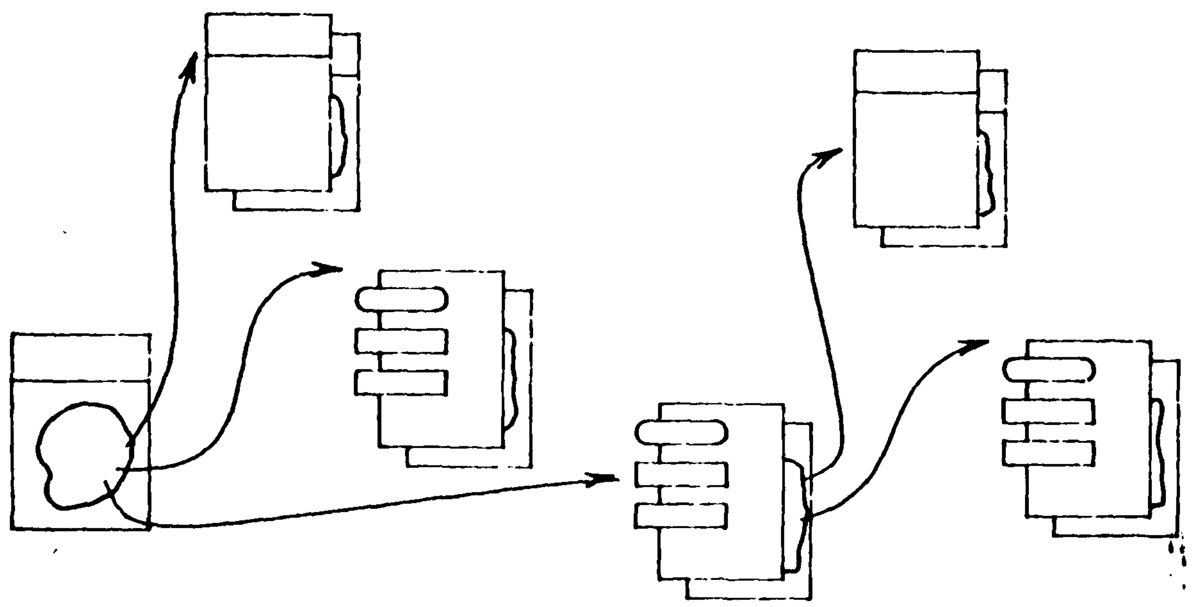
* Strong Typing
* Typing Structures
* Data Abstraction
* Tasks
* Exceptions
* Generics

SEPARATE COMPILATION

* Program units may be separately compiled

* Separate compilation is possible because of the separation of specification and body

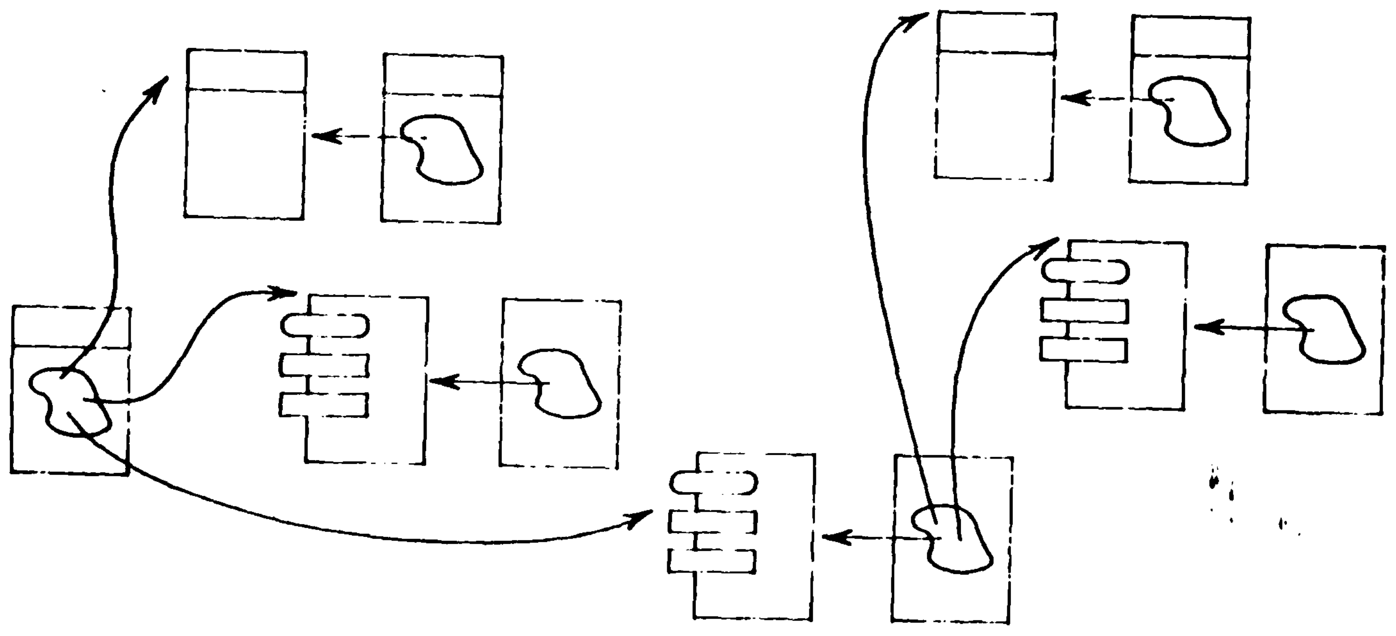
* A system is put together by referencing the specifications of other program units



SEPARATE COMPILATION

* A program unit's specification may be compiled separately from its body

* Realizes not only a logical distinction between architecture and implementation, but also a physical distinction



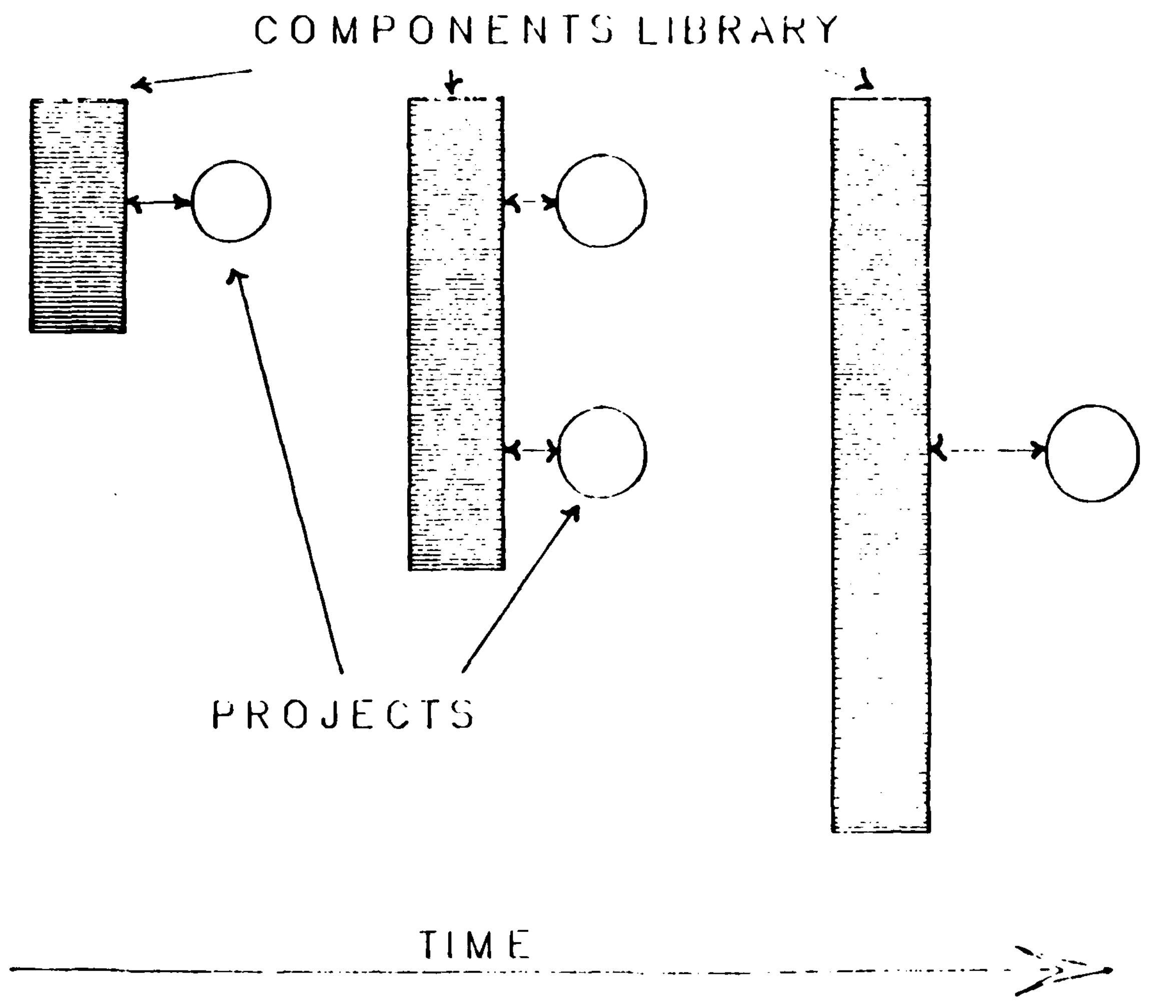
SEPARATE COMPILATION

* Allows development of independent software components

* Currently we all but lose the human effort going into software; it is disposable

* Separate compilation allows us to reuse components and keep our investment

SOFTWARE COMPONENTS



MAJOR FEATURES OF Ada

* Standardization	* Strong Typing
* Readability	* Typing Structures
* Program Units	* Data Abstraction
* Separate Compilation	* Tasks
$\boxed{\text{* Subprograms}}$	* Exceptions
* Packages	* Generics

DISCRETE COMPONENTS

* Allow a system to be composed of black boxes

* Provide clear, understandable functions

* Black boxes can be more effectively validated and verified

* Prevalent across engineering disciplines

SUBPROGRAMS

* A program unit that performs a particular action

  – Procedures
  – Functions

* Contains an interface (parameter part) mechanism to pass data to and from the subprogram

* The basic discrete component which acts like a black box

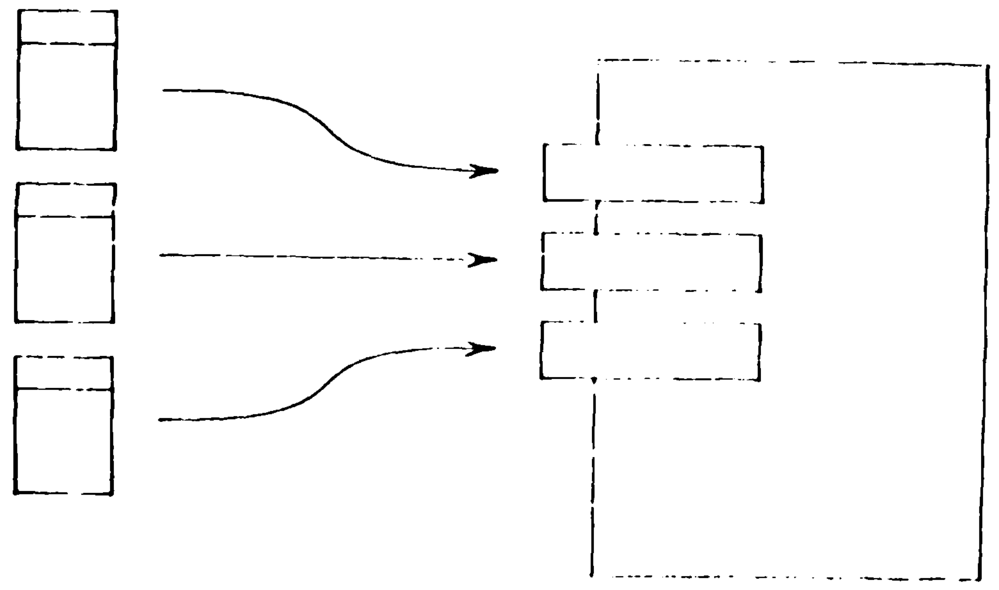
* Gives ability to express abstract actions

MAJOR FEATURES OF Ada

* Standardization
* Readability
* Program Units
* Separate Compilation
* Subprograms
* Packages
* Strong Typing
* Typing Structures
* Data Abstraction
* Tasks
* Exceptions
* Generics

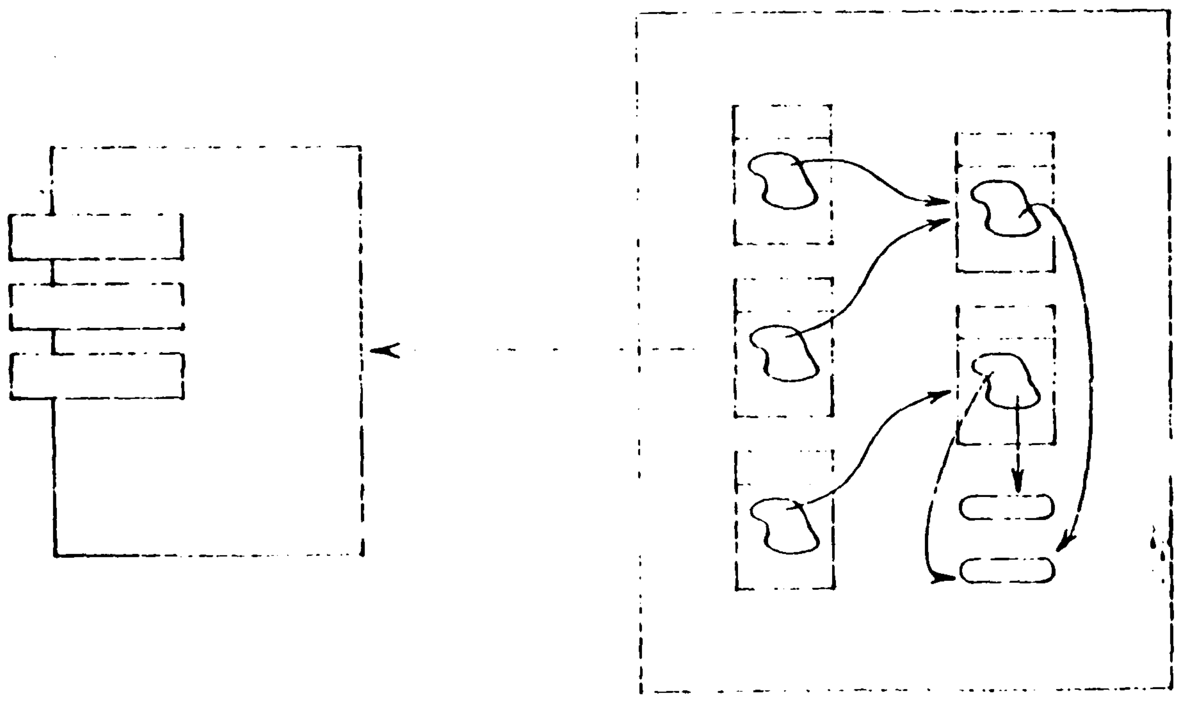
PACKAGES

* Program units that allow us to collect logically related entities in one physical place
* Allow the definition of reusable software components/resources
* A fundamental feature of Ada which allow a change of mindset
* An architecture—oriented feature



PACKAGES

* Place a "wall" around resources
* Export resources to users of a package
* May contain local resources hidden from the user of a package



PACKAGES

DIRECTLY SUPPORT:

* Abstraction
* Information hiding
* Modularity
* Localization

* Understandability
* Efficiency
* Reliability and safety
* Modifiability
* Correctness

MAJOR FEATURES OF Ada

* Standardization
* Readability
* Program Units
* Separate Compilation
* Subprograms
* Packages

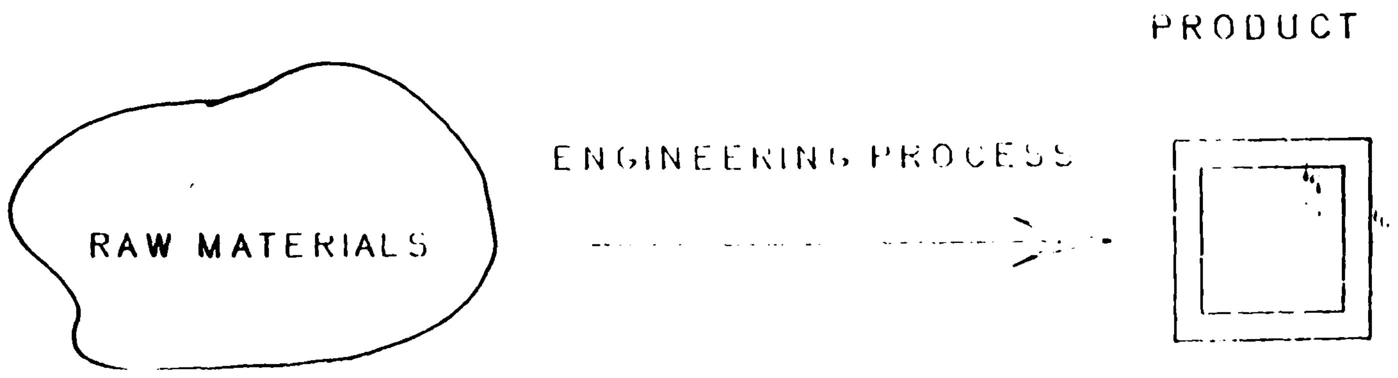
$\boxed{\text{* Strong Typing}}$

* Typing Structures
* Data Abstraction
* Tasks
* Exceptions
* Generics

THE RAW MATERIALS OF ENGINEERING

* All engineering disciplines shape raw materials into a finished product

* The materials and methods combine to define different disciplines



STRUCTURING RAW MATERIALS

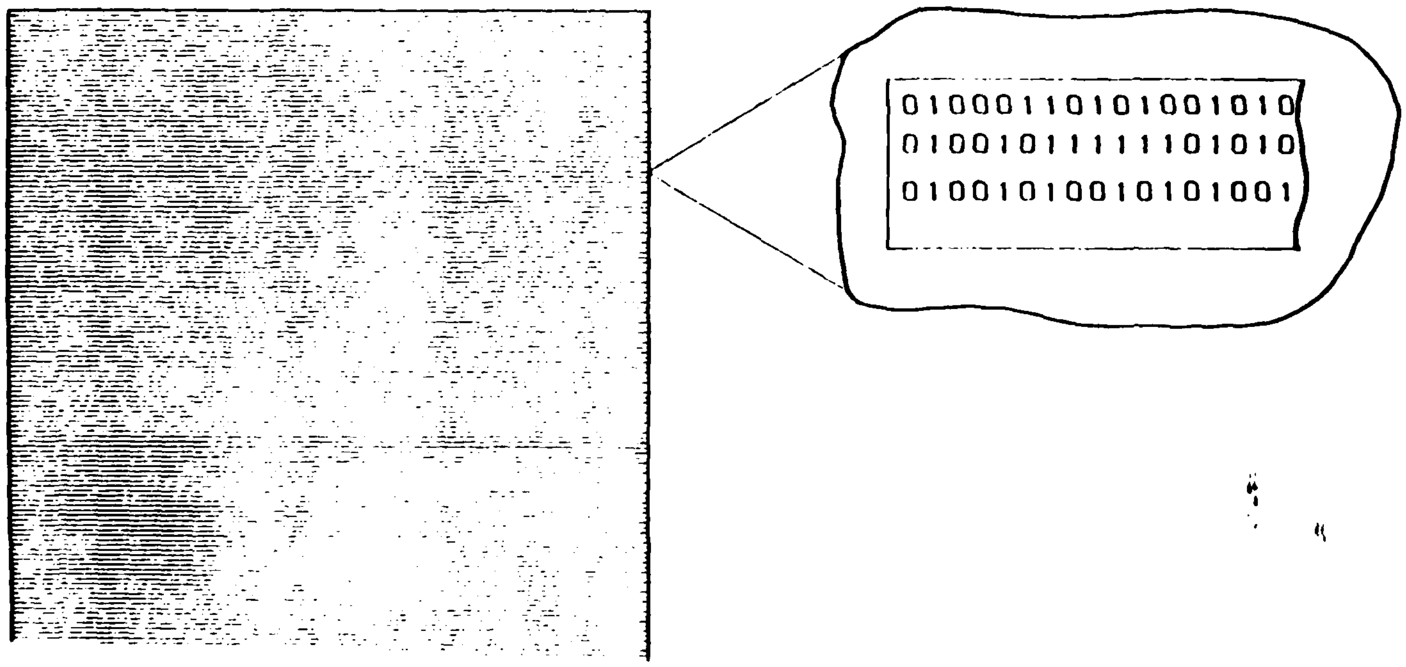
* There is a requirement to structure raw materials

  - To quantify
  - To manage
  - To test
  - To validate

* Methods of structuring vary across disciplines

SOME RAW MATERIALS OF SOFTWARE ENGINEERING

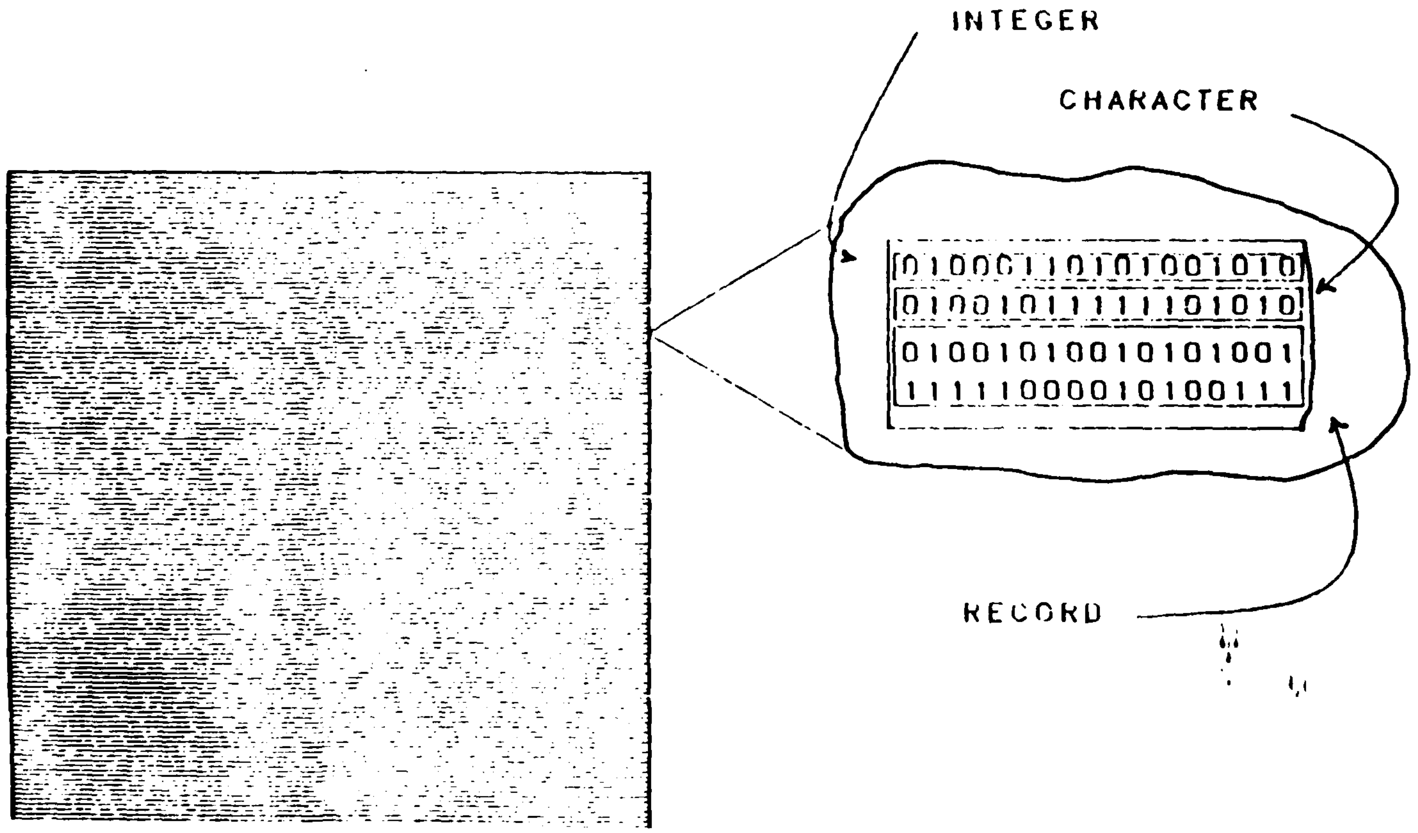
* Binary switches
* Computer memory locations
* Data



STRONG TYPING

* Defines structure of data (mapping)

* Enforces structure of data



STRONG TYPING

* Enforces abstraction of structure on data

* Increases confidence of correctness

* Increases reliability and safety

* Promotes understandability and maintainability

MAJOR FEATURES OF Ada

* Standardization
* Readability
* Program Units
* Separate Compilation
* Subprograms
* Packages

* Strong Typing
* $\boxed{\text{* Typing Structures}}$
* Data Abstraction
* Tasks
* Exceptions
* Generics

TYPING STRUCTURES

* Variety of problems requires a variety of structuring capabilities

* Ada provides a rich variety or types

TYPING STRUCTURES IN Ada

* Discrete data
- Enumeration
- Integer

* Real data
- Fixed point (absolute error)
- Floating point (relative error)

* Composite data
- Arrays (homogeneous)
- Records (heterogeneous)

* Dynamic data
- Access types

MAJOR FEATURES OF Ada

* Standardization
* Readability
* Program Units
* Separate Compilation
* Subprograms
* Packages
* Strong Typing
* Typing Structures
* Data Abstraction
* Tasks
* Exceptions
* Generics

DATA ABSTRACTION

* Combines primitive raw materials to form higher level structures

* Levels of abstraction

* Enforces an abstraction on a higher level structure

* Prohibits use of implementation details

* Promotes understandability

* Promotes modifiability

DATA ABSTRACTION AND PRIVATE TYPES

* Private types directly implement data abstraction

* Directly implement information hiding

```
package B_R is
   type NUMBERS is range 0..99;
   procedure TAKE ( A_NUMBER : out NUMBERS );
   function NOW_SERVING return NUMBERS;
   procedure SERVE ( NUMBER : in NUMBERS );
end B_R;
package body B_R is
   SERV_A_MATIC : NUMBERS := 1;
   procedure TAKE ( A_NUMBER : out NUMBERS ) is
   begin
      A_NUMBER := SERV_A_MATIC;
      SERV_A_MATIC := SERV_A_MATIC + 1;
   end TAKE;
   function NOW_SERVING return NUMBERS is separate;
   procedure SERVE ( NUMBER : in NUMBERS ) is
                                                        separate;

end B_R;
```

```
with B_R; use B_R;
procedure ICE_CREAM is
  YOUR_NUMBER : NUMBERS;
begin

  TAKE ( YOUR_NUMBER );
  loop
    if NOW_SERVING = YOUR_NUMBER then
      SERVE ( YOUR_NUMBER );
      exit;
    end if;
  end loop;

end ICE_CREAM;
```

```
with B_R; use B_R;
procedure ICE_CREAM is

   YOUR_NUMBER : NUMBERS;

begin

  TAKE ( YOUR_NUMBER );
  loop
    if NOW_SERVING = YOUR_NUMBER then
      SERVE ( YOUR_NUMBER );
      exit;
    else
      YOUR_NUMBER := YOUR_NUMBER - 1;
    end if;
  end loop;

end ICE_CREAM;
```

```
package B_R is

   type NUMBERS is private;

   procedure TAKE ( A_NUMBER : out NUMBERS );
   function NOW_SERVING return NUMBERS;
   procedure SERVE ( NUMBER : in NUMBERS );

private

   type NUMBERS is range 0..99;

end B_R;
```

```
with B_R; use B_R;
procedure ICE_CREAM is

  YOUR_NUMBER : NUMBERS;

begin

  TAKE ( YOUR_NUMBER );
  loop

    if NOW_SERVING = YOUR_NUMBER then
      SERVE ( YOUR_NUMBER );
      exit;
    else
      YOUR_NUMBER := NOW_SERVING;
    end if;
  end loop;

end ICE_CREAM;
```

```
package B_R is
   type NUMBERS is private;

   procedure TAKE ( A_NUMBER : out NUMBERS );
   function NOW_SERVING return NUMBERS;
   procedure SERVE ( NUMBER : in NUMBERS );

private
   type NUMBERS is range 0..99;
end B_R;
```

```
with B_R; use B_R;
procedure ICE_CREAM is

  YOUR_NUMBER : NUMBERS;

begin

  TAKE ( YOUR_NUMBER );
  loop

    if NOW_SERVING = YOUR_NUMBER then
      SERVE ( YOUR_NUMBER );
      exit;
    else
      YOUR_NUMBER := NOW_SERVING;
    end if;
  end loop;

end ICE_CREAM;
```

```
package B_R is

   type NUMBERS is limited private;

   procedure TAKE ( A_NUMBER : out NUMBERS );
   function NOW_SERVING return NUMBERS;
   procedure SERVE ( NUMBER : in NUMBERS );
   function "=" ( LEFT, RIGHT : in NUMBERS ) return
                                                    OOLEAN;

private

   type NUMBERS is range 0..99;

end B_R;
```

```
with B_R; use B_R;
procedure ICE_CREAM is

   YOUR_NUMBER : NUMBERS;
   procedure GO_TO_DQ is separate;

begin
   TAKE ( YOUR_NUMBER );
   loop
      if NOW_SERVING = YOUR_NUMBER then
         SERVE ( YOUR_NUMBER );
         exit;
      else
         GO_TO_DQ;
         exit;
      end if;
   end loop;
end ICE_CREAM;
```

MAJOR FEATURES OF Ada

* Standardization
* Readability
* Program Units
* Separate Compilation
* Subprograms
* Packages

* Strong Typing
* Typing Structures
* Data Abstraction
* Tasks
* Exceptions
* Generics

TASKS

* Program unit that acts in parallel with other entities
* Directly implements those parts of embedded systems which act in parallel
* Takes advantage of move toward parallel hardware architectures
  - Fault tolerance
  - Distributed systems
* Eliminates need to introduce additional complexity into a system

MAJOR FEATURES OF Ada

* Standardization
* Readability
* Program Units
* Separate Compilation
* Subprograms
* Packages
* Strong Typing
* Typing Structures
* Data Abstraction
* Tasks
* $\boxed{\text{* Exceptions}}$
* Generics

SOFTWARE RELIABILITY AND SAFETY

* Errors will occur

  – Hardware
  – Software

* Real time systems must be able to operate in a degraded mode

* Reliablity and safety must be engineered into a system

* Traditional languages lack specific features for dealing with errors and exceptional situations

EXCEPTIONS

* Deal specifically with errors and exceptional situations

* When an exception is raised processing is suspended and control is passed to an appropriate exception handler
    - Try again
    - Fix error
    - Propogate exception

* Increase reliability

* Reduce complexity

MAJOR FEATURES OF Ada

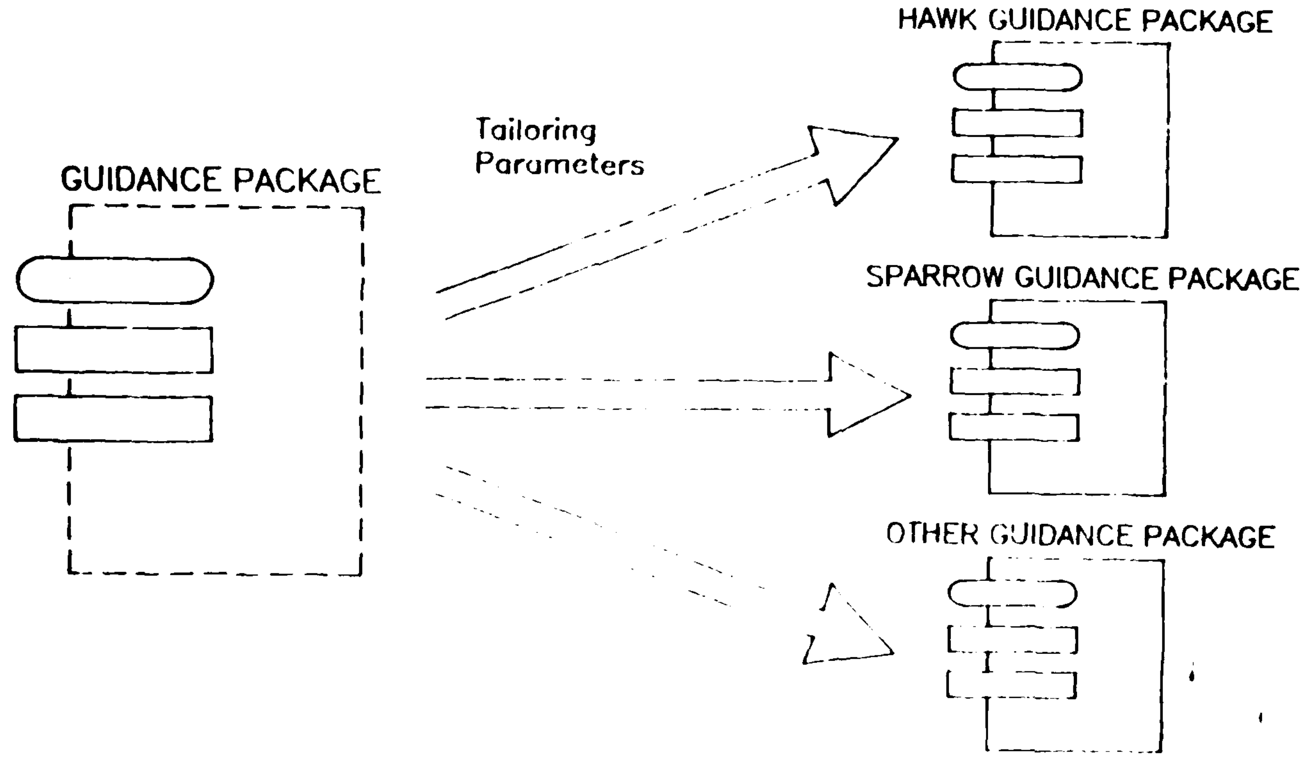
* Standardization
* Readability
* Program Units
* Separate Compilation
* Subprograms
* Packages

* Strong Typing
* Typing Structures
* Data Abstraction
* Tasks
* Exceptions
* Generics

GENERICS

* A generic is a tailorable template for a program unit

* Increases reusable software component capability by an order of magnitude



GENERICS

* Reduce size of program text

* Reduce need to reinvent the wheel

* Increase reliability by allowing reuse of known reliable components

OVERVIEW

* Rationale for development
* Capabilities and advantages
* Life Cycle application

SOFTWARE LIFE CYCLE

* Requirements analysis
* Preliminary design
* Detailed design
* Coding and unit testing
* Computer Software Component (CSC) integration and testing
* CSCI testing
* Maintenance

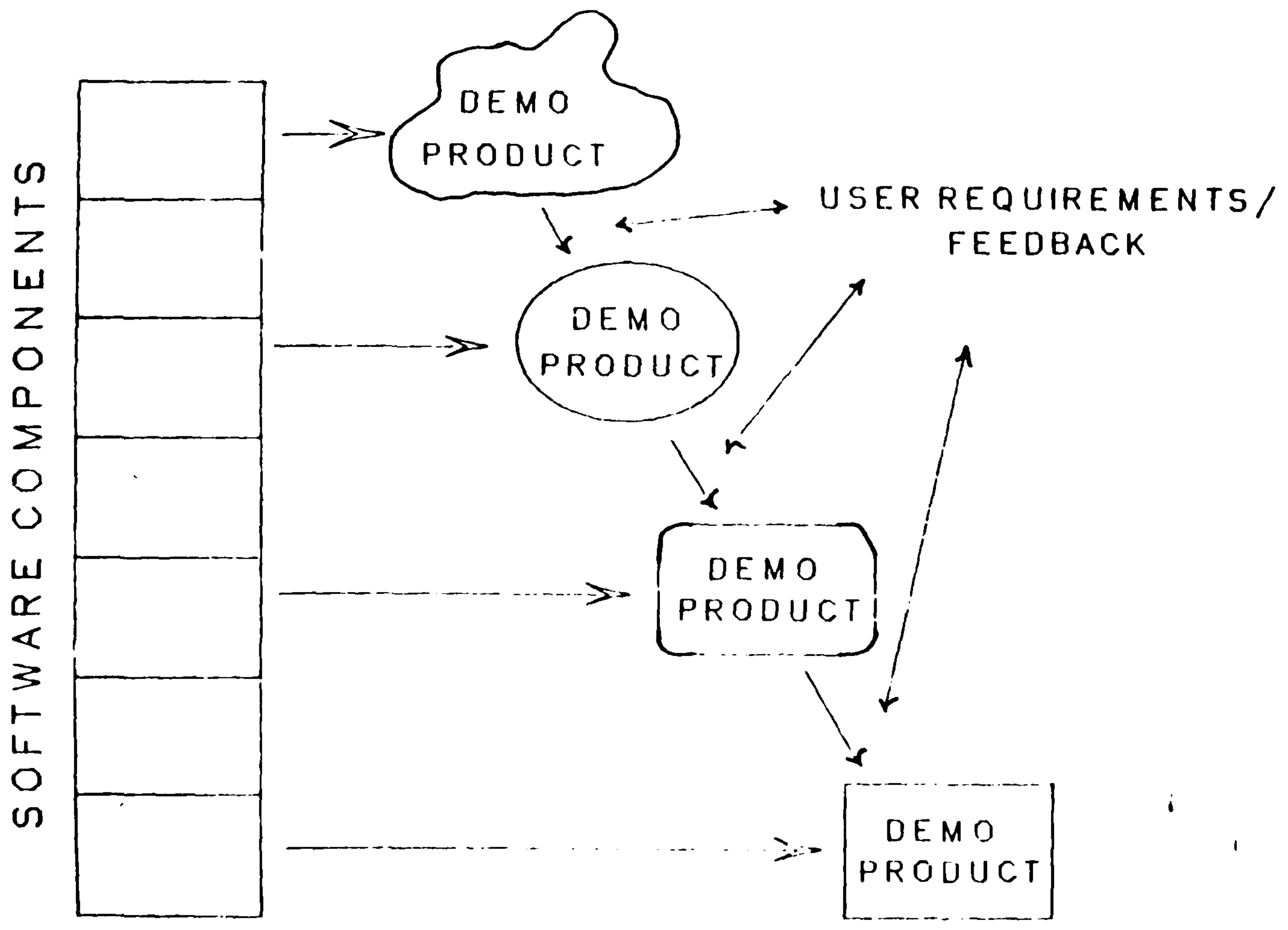
REQUIREMENTS ANALYSIS

* Standardization brings a much higher level of predictability

    – Ada language itself
    – Existing Ada software components

* Ada supports rapid prototyping very well

RAPID PROTOTYPING



DESIGN

* Ada features support architectural design

* Can actually express design in terms of PDL (Program Design Language)

    – Compilable
    – Allows other automated tool support

* Can enforce design through compilable PDL

* Ada supports varied methodologies

* Ada features reduce need to squeeze design into a programming language

CODING

* Ada features ensure original design is not violated

* Using PDL reduces amount of coding activity

* Readability of Ada code promotes productivity

TESTING

* The ability of Ada to support independent components allows more effective testing

* Exceptions allow "built–in" testing facilities

INTEGRATION AND TESTING

* Ada PDL ensures interfaces are correct

* More effective time can be spent testing the system rather than fixing integration errors

MAINTENANCE

* Readability makes maintenance much easier

* Proper software engineering using Ada will reduce maintenance costs

FILMED

3-8